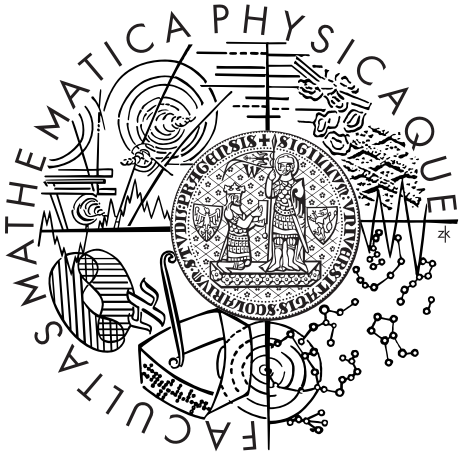
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Jiří Diviš

Visual Odometry from Omnidirectional Camera

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis:	Ing. Tomáš Svoboda, Ph.D.
Study programme:	Computer Science
Specialization:	Theoretical Computer Science

Prague 2012

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce: Vizuální odometrie ze všesměrové kamery

Autor: Jiří Diviš

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: Ing. Tomáš Svoboda, Ph.D.

Abstrakt: V této práci řešíme problém odhadu pohybu robota výhradně z obrázků pořízených ze všesměrové kamery, která je namontována na robotovi (vizuální odometrie [SF11, FS12]). V porovnání s hardware běžně používaným pro visuální odometrii, náš robot je specifický tím, že se pohybuje pomocí pásů a obrázky pořizuje pomoví všesměrové kamery s vysokým rozlišením a nízskou frekvencí snímkování (1 to 3 Hz). V naší práci se zaměřujeme na vysokou přesnost odhadů pohybu ve scénách, kde jsou objekty daleko od kamery. Toto je umožněno použitím všesměrové kamery. U tohoto typu kamer je známo že stabilizují odhad pohybu mezi pozicemi kamer, který je špatně podmíněn u kamer s malým zorným polem. Dále využíváme faktu že kamera má nízskou snímkovací frekvenci k tomu, abychom využily uvolněných výpočetních zdrojů pro zpracování obrazu ve vysokém rozlišení. Pro odhad pohybu kamery používáme metodu založenou na detekci rohů. K vůli možnosti velké vzájemné rotace kamer mezi snímky jsme nuceni použít metodu párování rohů namísto trackingu.

Klíčová slova: vizuální odometrie, spherická aproximace, odhad pohybu kamery.

Title: Visual Odometry from Omnidirectional Camera

Author: Jiří Diviš

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Ing. Tomáš Svoboda, Ph.D.

Abstract: We present a system that estimates the motion of a robot relying solely on images from onboard omnidirectional camera (visual odometry [SF11, FS12]). Compared to other visual odometry hardware, ours is unusual in utilizing high resolution, low frame-rate (1 to 3 Hz) omnidirectional camera mounted on a robot that is propelled using continuous tracks. We focus on high precision estimates in scenes, where objects are far away from the camera. This is achieved by utilizing omnidirectional camera that is able to stabilize the motion estimates between camera frames that are known to be ill-conditioned for narrow field of view cameras and the fact that low frame-rate of the imaging system allows us to focus computational resources on utilizing high resolution images. We employ feature based-approach for estimation camera motion. Given our hardware, possibly high ammounts of camera rotation between frames can occur. Thus we use techniques of feature matching rather than feature tracking.

Keywords: visual odometry, spherical approximation, camera motion estimation.

Contents

Introduction

The topic of this thesis is to design a visual odometry (VO) system for the robot on the photograph in Figure XXX. The robot already has the capability to perform odometry using combined sheel encooder data and internal measurment unit data (XXX citace). The laser-based odometry for the robot is also beeing developed by others. The motivation behind having VO for the robot is that it is assumed that it will have greater precission and that it will work in places where the other odometries do not. Concreatly leaser-based odometry suffers from the fact that laser range is approx. 10 m. The combined wheel and internal measurement unit odometry suffers from wheel slip and rough terrain conditions (high vibrations).

Concering the robot hardware, it is equiped with Ladybug 3 camera, which is omnidiractional camera composed of wide-angle 6 perspective cameras. The on-board computer is capable of producing either 3200 x 1600 panoramatic image, or 6 1200 x 1600 individual images at framerate of 2 frames per seconf. The robot is propelled by caterpilar track and that it has flipper that can lift front (or back) of the wehicle.

Given the hardware constrains and available odometries on the robot, the design goals of our VO are as follows.

- VO has to be able to reliably estimate motion in an environment, where there is wery little or no close objects in it. It does not have to work in environments where there are only close objects.
- VO has to be able to work with slow framerates (compared to standart 24 hz of standard cameras), but it can take much longer to process frames.
- VO should be able to achieve high accuracy. More concreately, has to be able to utilize omnidirectional image, which has been proven in theory and practice to be far superior to that of standard perspective camera [**?**].

To further categorize our work in context of VO [SF11, FS12]. Our systems belons to the class of feature-based systems, where features corespondences are determined by matching rather than tracking. Concerning our camera model, we use modyfied spherical central projection model. We use standard 5-point motion estimation algorithm in RanSaC scheme, the estimates are than jointly refined over multiple camera frames. We do not assume any prior knowledge about the motion performed by the robot. We do not solve calibration problem for our camera model. Finally, to evaluate our work, we qualitatively determine how well robot closes loops and how it behaves in different environments.

XX related work.

1. System Analysis and Design

1.1 Overview

We would like to give an overview of of major components of our system and explain how interact with each other to form the desired behaviour. Division into the modules coorespond with the actual implementation. It is also stable enough to survive severe design changes and thus the intent of this section is to provide context for the discussion of both analysis of the VO problem and actual implementation of individual parts of the system.

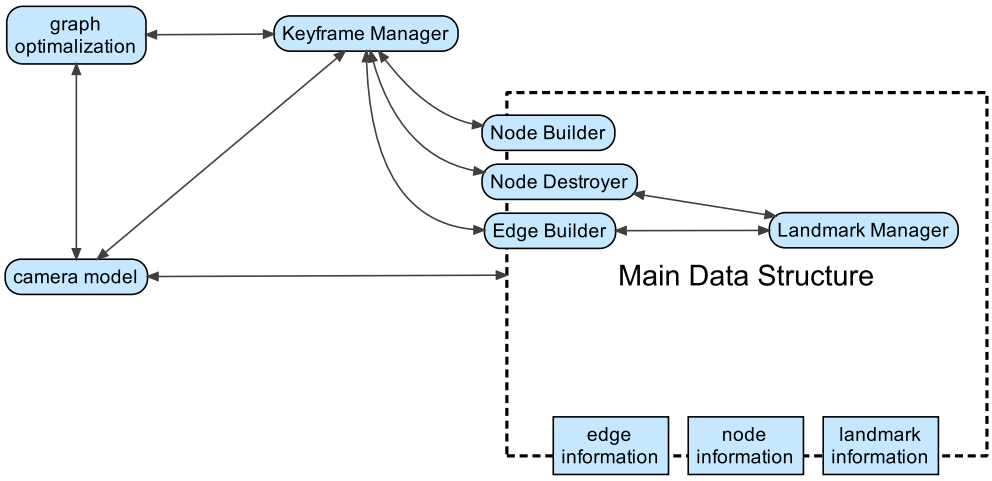


Figure 1.1: Overview of system modules and their interactions. Round-edged boxes represent input/output module interfaces and sharp-edged boxes represent output-only interfaces. Arrow indicates that source module uses target module.

1.1.1 The Main Modules of the System

We describe main modules of the system, which are illustrated in Figure 1.1. The main data structure is described in terms of its function with relation to the rest of the top-level modules.

Information Maintained by the Main Data Structure

We represent data that are required for operation of the VO system as an oriented graph G, with additional constraint $(a, b) \in G \Leftrightarrow (b, a) \in G$. Nodes represent frames and edges represent constrains between nodes. Part of this representation is also a landmark, which represents locations of points of iterest in the scene.

A node of the graph represents a pose of the camera in space as a rigid body transform from origin of the odometry coordinate frame to the coordinate frame of the camera represented by the node. An image of the scene that is taken from this pose is also represented by node — as an array of selected *image features* detectable from this pose (or equivalently, this *viewpoint*). *Image feature* (or simply *feature*) is well detectable image pattern in an image that can be redetected from images taken from nearby poses. To be more precise, the feature

array is are represented as an array of *keypoints* and the corresponding *descriptors* for the keypoints. *Descriptors* contain parts of the feature representation which is useful only for feature matching, while *keypoint* represents information about features that are also useful elsewhere (like image coordinates, or strength of the feature detection response for that feature).

An edge of the graph represents two things. First, it represents transform from the edge source camera pose coordinate frame to the edge target camera pose coordinate frame. Second, it maintains an association of pairs of features in the edge source and edge target that are images of the same *landmark*. *Landmark* is a part of the scene that gives rise to a feature in an image (i.e., it is coimage of the feature). They are represented as 3D points.

The landmark structure represents 3 things. First, it maintains an association of features-node pairs with landmarks from which they are observed. Second, it maintains whether a meaningful 3D estimate of landmark can be computed. Third the 3D estimate of landmark position (if applicable). The issue of maintnane of Landmark structure is discussed in section 1.4.

Interface for the Manipulation of the Main Data Structure

The interface consists of routines which create nodes (*Node Builder*), create edges (*Edge Builder*) and delete nodes with consequence of deleting edges (*Node Destroyer*). The addition and deletion of edges invokes changes in the landmark substructure, which is performed by *Landmark Manager* (section 1.4).

Main Purpouse of Node Builder is to process input image into array of features and its corresponding descriptors. After invocation of Node Builder, the processed image is discarderd.

Edge Builder initiates process where most of the work of our VO system is done. Edge Builder parameters are source and target nodes that are to be connected by oriented edge. It also accepts the type of edge parameter which determines what kind of estimation is done. Further division of Edge Builder into submodules is discussed in subsection 1.1.2. The arrangement of the submodules that form Edge Builder is than discussed in subsection 1.1.3.

Node Destroyer takes the node that is to be deleted as a parameter. The only interesting issue with this operation is that as a consequence of the node deletion, landmark structure needs to be altered to reflect deletion of features and corresponding landmark-feature associations. This is discussed as a part of Landmark Manager in section 1.4.

Graph Optimalization

Graph optimalization module utilizes the featere-landmark constraints that arise from feature measurements in an image and initial pose estimates of nodes and landmarks to jointly optimalize estimates of the camera poses. This is a computationally expensive step and it relies on Keyframe Manager to maintain such graph structure in Main Data Structure that keeps only meaningful constraints. Graph optimalization module, that employs nonlinear optimalization method, called *bundle adjustment (BA)* in context of computer vision, is discussed in section 1.6.

Keyframe Manager

Keyframe Manager is the top level module that directly or indirectly controls all other modules and is an interface of the whole system. Its function is to issue appropriate commands to Node Builder, Edge Builder and Node Destroyer so as to obtain graph structure and thus also node, edge and landmark information that is proper for BA. After BA, pose information of active node is outputed to fulfill VO task. *Active node* is the node that correspond to the last camera image recieved and processed by Node Builder. The nodes in the graph that are not used in BA are of no further use for VO and thus can be deleted by Node Destoyer. Nodes kept for BA, with the exception of the active node, are commonly called *keyframes* in the context of BA (thus the name Keyframe Manager). Keyframe manager is the topic of section 1.5.

Camera Model

Camera model module is used heavily in almost all modules. Given pose in the scene, its function is to map pixels in the image to the parts of scene that determine the value of this pixel and vice versa. Camera model is the subject of subsection 1.2.2.

1.1.2 Individual Subcomponents of Edge Constructor Module

We describe submodules from which Edge Builder function is composed. They are described in terms of required input and their output. Because these modules use only information that is part of Main Data Structure, the input requirements also specify which parts of MDS have to be filled for use of this module. The proper order of execution of these modules is then subject of subsection 1.1.3 and Figure 1.7.

Feature Selection

intput Array of keypoints and its corresponding feature descriptors (e.g. such as in a node of the Main Data Structure).

output Array of filtered keypoints and its corresponding array of descriptors. Feature can be selected by non-maxima supression (subsection 1.3.2) of feature detection response strength given required maximum acceptable feature density per unit squared of image coordinate (subsection 1.3.2).

Unguided Matching

input Array of keypoints that are to be matched and array of corresponding feature descriptors. This is useful to reduce computational requirements of subsequent steps.

output Set of pairs (i, j), where i is an index of a feature in the array of features originating from image of source node and j is an index of a feature in the array of target node. It is assumed that if features are paired, that there is reasonable probability p that the paired features are images of the same landmark. Requirements on p are defined by robust estiamtion module that uses the output, typically p ranges from 0.2 to 0.8).

The task performed by this module is called *feature matching* in general and the pairs (i, j) are called *correspondences*. In more specific terms, the kind of feature matching employed by this module is called *unguided matching* in the sense that no geometric constrains are exploited in matching process and only feature information is used. Unguided matching is the topic of subsection 1.3.3.

Guided Matching

input Array of keypoints that are to be matched and array of corresponding feature descriptors. Rigid body transform between coordinate frames of the two nodes involved in matching. Optionally, landmark-feature association for features in one of the nodes and 3D coordinate of the landmarks involved.

output For *guided matching*, it is qualitatively same as for unguided matching. Guided matching is the topic of **??**.

Robust Model Estimation

[1]

input Feature-feature correspondences from two nodes with reasonable number of outliers. RanSaC is used for model estimation, and thus outlier to inlier ratio depends on model size, which in turn determines number of iterations required. Computational requirements of various model estimation algorithms are leading factor in time complexity of one iteration.

output The feature-feature correspondences that fit the model and the estimated model — rigid body transform beetween node coordinate frames. Robust model estimation the topci of subsection 1.2.1

Model Refinement

input Rigid body transform estimate (initial model estimate) between the two nodes involved. Feature correspondences that fit the model estimate.

output Refined model estimate (typically using all inlier feature correspondences).

Outlier Removal Given Estimated Model

input Rigid body transform estimate (model estimate) between the two nodes involved.

[1] XXX fix code — landmark management sould run between local and global matching

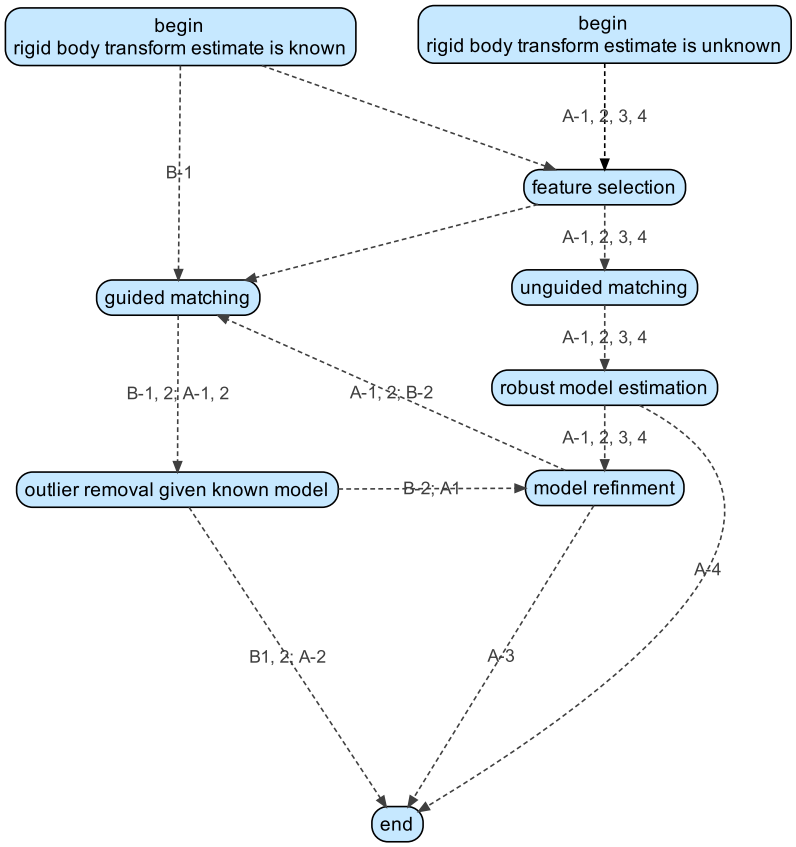


Figure 1.2: Possible arrangements of modules to fulfil Edge Builder interface.

output Feature correspondences that fit the model estimate.

1.1.3 Edge Construction Component Types

We present arrangment of the components from the above section which produces possible implementations of Edge Builder. These arrangement are illustrated in Figure 1.7. They fall into two major types. First, the case where the rigid body transform is not known. Second, the case where the rigid body transform between nodes is already known (e.g. from internal measuremnt unit).

Unknown Rigid Body Transform

The most complete solution uses the modules in the following order: **feature selection** (to reduce number of features to tractable levels for unguided matching), **unguided matching** (to find feature corespondences), **robust model estimation** (to remove outliers and estimate the rigid body transform), **model refinement** (compute more precise estimate using all data), **guided matching** (to increase number of correspondences), **outlier removal given known model**. This corresponds to path A-1.

Other options include skipping **model refinement** (this is the case that we actually implement) and/or skipping the **guided matching** with **outlier removal given known model** (if we have enough correspondences). This corresponds to paths A-2, A-3 and A-4.

Known Rigid Body Transform

This case is not required for VO to work (and we have not implemented it), but it can help significantly reduce computation time when the rigid body transform estimate is already available. This is useful if VO is combined with e.g. wheel odometry or internal measuremnet unit. It can be also useful in keyframe manager for creating additional constrains in the pose-graph or in case when only rotational part of the estimate is known (see **??**.

Desirable arrangement of modules depends on the quality of the initial rigid body transform estimate. If it is fine, only two steps are necessary: **guided matching** and **outlier removal given known model** (B-1). For more coarse estimates, thw following steps would become necessary: **guided matching**, **outlier removal given known model**, **model refinement**, **guided matching** (again for more matches), ... (B-2).

1.2 Estimating Relative Rigid Body Transforms Between Cameras

In this section, we discuss the problem of computing rigid body transform between two node coordinate frames from set of feature-feature correspondences. We discuss various solvers that solve for the transform using featur-feature correspondences or landmark-feature correspondences. All these solvers require camera model to determine for each pixel coordinate in set of correspondences oriented vector indicating the ray that is projected to the pixel with the specified coordinate. We thus also discuss camera models. Minimal number of correspondences required by the solvers is low (3–6). We use random sample consensus (RanSaC) method to remove outliers.

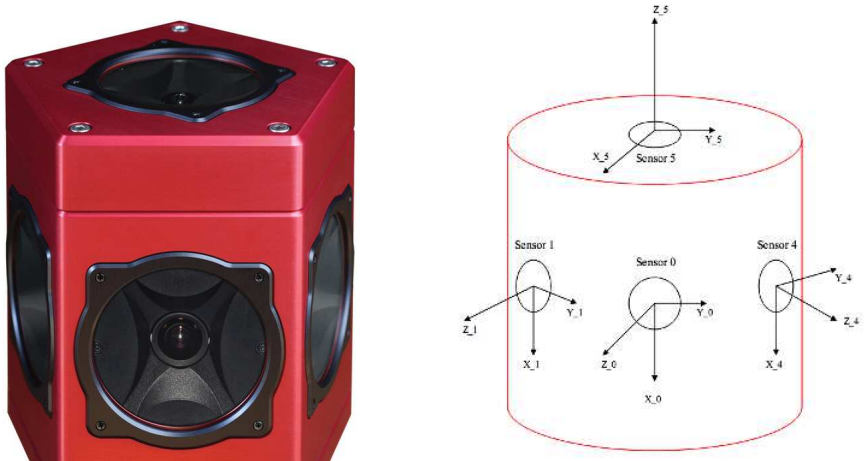
1.2.1 Model Estimation in Presence of Outliers

Number of RanSaC Iterations Required

1.2.2 Camera Models

We define *camera model (CM)* using common geometric abstraction used in leterature [SRT+11, MSKS10]. Camera model determines which regions of space are imaged on what regions of cameras imaging surface. The regions of scene that affect a point on the imaging surface are modelled by rays. We parametrize *Ray* as (B^f, B^i), where B^f is some point on the ray in 3D space and B^i is some direction vector of the ray. *Camera model (CM)* is fully determined by the follwoing:

- space of its imaging surface $\mathcal{I}$
- the space $\mathcal{R}$ of rays sampled by the camera.
- camera parameters p (*intrinsic camera parameters*).
- one to one map $\pi_p \colon \mathcal{R} \to \mathcal{I}$ between the space of imaging surface $\mathcal{I}$ and of the camera and the space of rays sampled by the camera $\mathcal{R}$.



(a) Ladybug camera.

(b) Ladybug camera. Orientation of optical axes of individual prespective cameras.

Figure 1.3: Ladybug Camera. Images taken from `www.ptgrey.com`.

The function π_p is called *forward projection* and the function π_p^{-1} is called *back-projection*.

Camera models can be divided into central and non-central. Camera model is *central camera model (CCM)*, if all the rays intersect at one point (called *optical center*). Otherwise it is *non-central camera model (NCM)*. Consequently rays in central CCM model can be represented using direction vectors B^i only.

In order for function π_p to be determined, camera parameters have to be determined. Process of estimating camera parameters (*calibration*). Calibration will not be discussed

Designing CM and calibrating it is not subject of this theses, we assume that p is known and we signify this with p in π_p. We were provided few options for CM. Relevent aspects of these are very briefly discussed in the section bellow.

Non-central Model of Ladybug Camera

Ladybug camera consist of six standard cameras with fixed intrinsic parameters adhering to the ideal pinhole camera model with polynomial distortion, similar to one described in [MSKS10, Chapter 3]. Five of these cameras are arranged so that their optical axes lie in one plane and meet at one point O_s. Sixth camera has optical axis perpendicular to the plane and also intersects the other optical axes at point O_s. The arrangement of cameras in Ladybug camera is depicted in Figure 1.3b, while the picture of the real camera is in Figure 1.3a. The cameras have wide field of view so that all rays with direction vectors pointing above (as defined by Figure 1.3) the plane defined by the optical axes of the first five cameras are sampled by at least one of the cameras (possibly more).

The optical centers of the indivividual cameras do not coincide with O_s. Therefore, if we wish to model Ladybug camera by single camera model precisely, that camera model cannot be central camera model. We designate the optical centers of the indivisual cameras as $O_0, \ldots, O_5$ and describe appropriate camera model for ladybug camera as follows.

- The virtual imaging surface $\mathcal{I} = \bigcup_{j=0,\dots,5} \mathcal{I}_j$ of Ladybug camera consists of the six parts that correspond to imaging planes $\mathcal{I}_j$ of the individual cameras satisfying pinhole CM.

- The space of rays sampled by the Ladybug camera is

$$\mathcal{R} = \bigcup_{j=0,\dots,5} \{(O_j, B_j^i) | B_j^i \in \mathcal{B}_j\}$$

  where $\mathcal{B}_j$ is the set of rays sampled by the camera with optical center O_j.

- The Ladybug camera parameters p consist of relative positions of O_j and O_s and the parameters p_j of the individual pinhole cameras. XXX cite calibration method.

- The forward projection $\pi_p \colon \mathcal{R} \to \mathcal{I}$ is defined by:

$$\pi(O_j, B_j^i) = \pi_{p_j,j}(B_j^i)$$

  and back-projection is thus defined by:

$$\pi_p^{-1}(I_j) = (O_j, \pi_{p_j,j}^{-1}(I_j))$$

Spherical Approximation for Ladybug Camera

Conviniet model applicable to all central projection cameras is *spherical camera model* which is defined as follows:

- The virtual imaging surface $\mathcal{I}$ is a unit sphere.

- Space of rays $\mathcal{R}$ samples visible rays B^i with optical center O_s.

- The forward projection $\pi_p \colon \mathcal{R} \to \mathcal{I}$ is defined by

$$\pi_p(O_j, B^i) = (x, y, z)$$

  where (x, y, z) is the direction vector of a ray B^i such that $\|(x, y, z)\| = 1$.

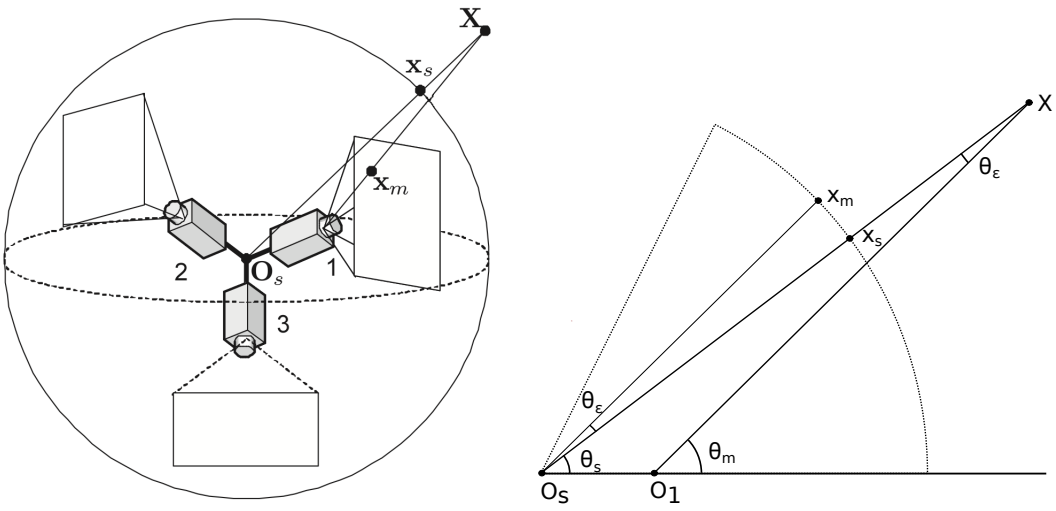
- The back-projection π_p^{-1} is identity.

If we use notation introduced in the definition of Ladybug CM, than in *spherical approximation* of Ladybug camera the optical center O_s is assumed to coincide with optical centers O_j. Consider a ray $(O_s, B^i) \in \mathcal{R}$ such that there exists camera j and ray $B_j^i \in \mathcal{R}_j$ for which $B^i = B_j^i$. In the approximation the rays rays B^i and B_j^i sample sample same part of the scene and thus

$$c(\pi_p(B^i)) = c_j(\pi_{p_j,j}((O_j, B_j^i)))$$

where c, c_j are functions that map point on imaging surface to its brightness value.

In the following, we atempt to characterize error induced by the spherical approximation. We follow the derivation presented in [KHK10]. Consider a landmark with coordinates X that is imaged by ray B_j^i of camera j. In coordinate frame of Ladybug CM, this corresponds to the ray (O_j, B_j^i). Using a spherical



(a) Error in ray direction induced by spherical approximation. Image adapted from [KHK10].

(b) Detail of spherical approximation error (see the text).

Figure 1.4: Spherical approximation.

CM, we designate the ray corresponding to the landmark X as B^i. Such situation with $j = 1$ is drawn in Figure 1.4a. The rays (O_s, O_j), B^i and (O_j, B^i_j) determine a plane, which is drawn in Figure 1.4b. In the following, the approximation error is characterized using angle between rays (O_j, B^i_j) and B^i which is designated as θ_ϵ. This is then related to angle θ'_ϵ which is maximum angle such that error of approximation on the imaging surface of spherical CM, which is determined by $e_j := \|x_s - x_m\|$, does not exceed one pixel.

The derivation of θ_ϵ follows. We define the distance of landmark X from optical center O_s as d_r and we define t_j as $\|O_s - O_j\|$. For Ladybug CM, $t_j = 0.042\,\mathrm{m}$ for horizontal cameras (cameras $0 - 4$) and $t_5 = 0.062\,\mathrm{m}$ for the top camera (camera 5). Using the law of sines, we get

$$\frac{t_j}{\sin\theta_\epsilon} = \frac{d_r}{\sin(\pi - \theta_m)}$$

noting that $\theta_m \leq \pi$, we have $\sin\theta_m = \sin(\pi - \theta_m)$ an thus

$$\frac{t_j}{\sin\theta_\epsilon} = \frac{d_r}{\sin\theta_m}$$

For minimal angle θ'_ϵ between the rays intersectiong pixel boundaries, we get following condition on d_r, t_j and θ_m

$$\sin\theta'_\epsilon \geq \frac{t_j}{d_r}\sin\theta_m$$

This means that for specified error of approximation θ'_ϵ, maximal angle $\theta'_m = \max\theta_m$ and distance $\|O_i - O_s\| = t_j$, the minimal distance of the scene d_r has to be:

$$d_r \geq \frac{t_j \sin\theta_m}{\sin\theta'_\epsilon} \tag{1.1}$$

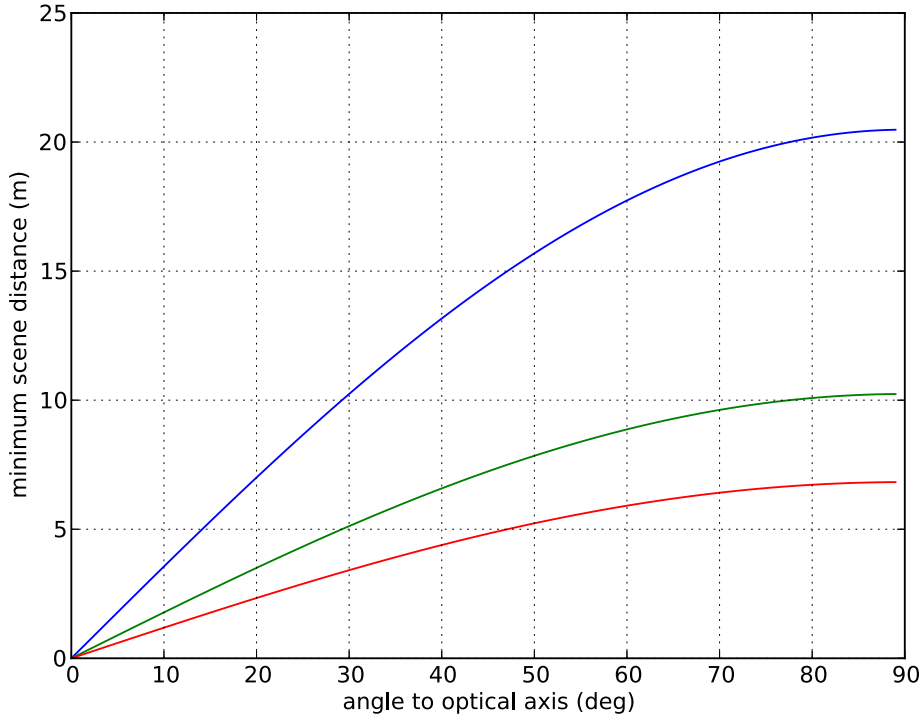


Figure 1.5: Plot of minimum disctance to the scene as a function of the angle between optical center and imaged ray. The plots correspond to errors of one, two and three pixels.

The plot that plots d_r a s function of θ_m, which is determined by Equation 1.1 is presented in Figure 1.5.

We determine θ'_ϵ as minimum of (f_w/w) and (f_h/h), where these quantities have following meaning (with concreate values for Ladybug CM.

- w is image width in pixels. $w = 1200\,\text{px}$.
- h is image height in pixels. $h = 1600\,\text{px}$.
- f_w is horizontal field of view of cameras (same for all of them). $f_w = 120°$.
- f_h is vertical field of view of cameras (same for all of them). $f_h = 160°$.

With these values, error is guaranteed to be within 1 px, if scene distance d_r to optical center O_s is less than 20 m (for horizontal cameras). See Figure 1.5

Panorama for Spherical Approximation

[2] Camera model, where imaging surface is a panorama XXX jak se tomu rika.

- imaging surface is defined by $\mathcal{I} = (\theta, \varphi)$, where $\theta \in (-\pi, +\pi)$ and $\varphi \in (-\frac{\pi}{2}, +\frac{\pi}{2})$.

[2] neznam vhodny nazev XXX

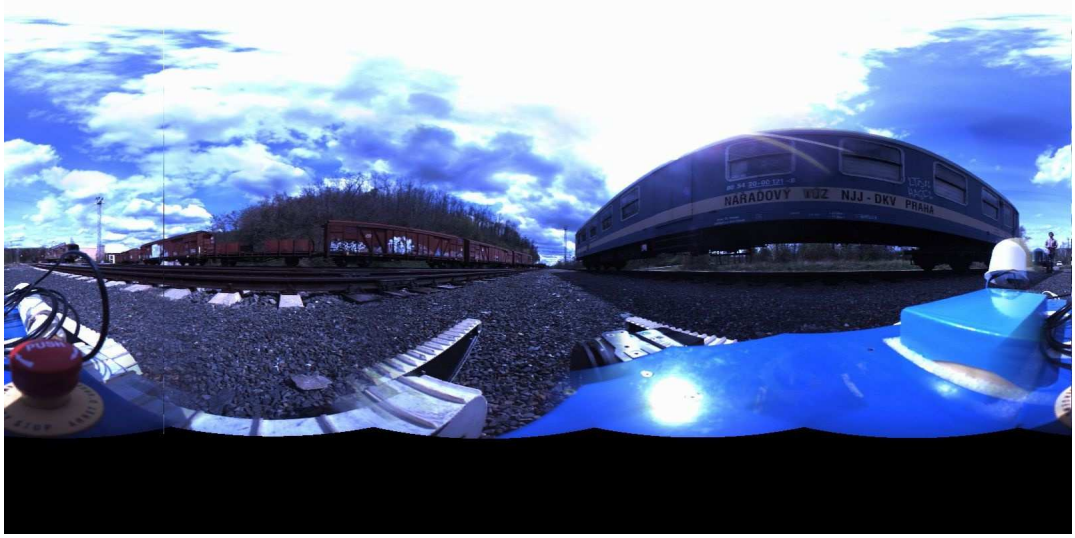


Figure 1.6: Panoramatic image constructed from spherical approximation.

- the space $\mathcal{R}$ is defined as:

$$\mathcal{R} = \{(0, (x, y, z)) | \|(x, y, z)\| = 1 \wedge |z| \neq 1\}$$

- intrinsic camera parameters p are the same as for spherical approximation.
- The projection functions correspond to conversion between spherical and cartesian coordinates. More specifically, forward projection $\pi_p \colon \mathcal{R} \to \mathcal{I}$ is defined as

$$\pi_p((\theta, \varphi)) = (\text{atan2}(-x, z), \text{atan2}(y, \sqrt{x^2 + z^2}))$$

  and back-projection is

$$\pi_p^{-1}((x, y, z)) = (\sin(\theta)\cos(\varphi), \sin(\varphi), \cos(\theta)\cos(\varphi))$$

1.2.3 Model Estimation

Motion Estimation and Scale estimation

Essextial matrix XXX. Problem statement XXX. n-point XXX. Error function XXX. Solvers XXX.

Pose Estimation

Problem statement XXX. PnP XXX. Solvers XXX.

Non-Central Models

Vague problem statement XXX. Solvers XXX.

1.2.4 Implemented Solution and Concluding Remarks

We implemented motion estimation using RanSaC scheme with essential matrix for CCM as the model that was estimated. We used 5-point minimal solver for CCM described in [LH06]. As an error function for data point we use the one described above. The method for scale estimation and method from extraction rotation and direction vector from essential matrix is also implemented as described above.

In the following, we explain our decision to use spherical approximation and our choice of method for relative motion estimation. The discussion of choice of panorama as virtual imaging surface is in section 1.3.

Choice of Central Camera Model over Non-central Camera Model

In [KHK10] there is an experimental comparison in simulated environment of the following camera models in motion relative motion estimation scheme very similar to ours:

1. Spherical approximation of an arrangement of 3 cameras similar to arrangement of cameras in Ladybug CM.

2. Single camera that can be modelled as pinhole CM.

3. An arrangement of 3 cameras as in 1., except that their optical centers coincide is a single point.

The error in imaging surface coordinates induced by approximation in 1. was already discussed in subsection 1.2.2. Using terminology from that Section, if $d_r > 20\,\mathrm{m}$, then the error introduced by the spherical approximation coincides with feature tracking error. Infering from experiments presented in [KHK10], it is reasonable to expect that spherical approximation will not behave much worse if d_r is as much as 10 times smaller.

This does not prohibit us to use spherical approximation, because of the fact that our design goals consider acceptable the possibility that the algorithm will not work in such small scene depths as it is expected that laser based odometry performs acceptably in such scenes.

There are other advantages to using spherical approximation. Namely that it is simpler non-central NCM and there are more methods and implementations for 5-point solvers. Additionally, as explained in subsection 1.2.1, using 5-point solver rather than 6-point solvers results in less iterations of RanSaC. Finally, given the fact that we use BA fo refine our estimates, one could use motion estimation results obtained using spherical approximation to initialize BA which uses exact CM. In such scheme distance of cameras from O_s would probably have to be considered as a parameter, because the distance of the cameras from O_s are probably too small for reliable scale estimates and scale is thus unknown.

Method Used for Model Estimation

Given the fact that thare exist solvers usable in RanSaC scheme for both motion estimation and pose estimation problems with both NCM and CCM case. We opted for RanSaC as a method for robust estimation as recomanded in [FS12].

Regarding our choice of relative position estimation method. The implementation of motion estimation using 5-point algorithm is a must, because at various points in the VO process, 3D estimates may be unavailable. If 3D landmark estimates are available, pose estimation would probably offer better initialization of scale estimate for BA. Pose estimation is commonly used in VO [TPD08, NNB06].

It has already been established in the Introduction, that omnidirectional vision is far superior to single camera satisfying pinhole CM. The following is motivated by the fact that in our experiments and experiments of others, it was found that scale is very difficlut to estimate as it is suceptible to drift [SMD]. From the presentation of available solvers for CCM and NCM, it seem that thare are algorithms of comparable quality available for severly NCM. Such algorithms have an advantage of beeing able to directly (i.e. from two camera nodes) estimate scale given that the camera optical centers are far enough apart (which ours are not). Such camera would wery likely produce very superior scale estimates compared to stereo vision, especially for large vehicles.

1.3 Feature Detection and Matching

1.3.1 Feature Detectors and Feature Descriptors

1.3.2 Fast Feature Lookup

1.3.3 Unguided Matching

1.3.4 Guided Matching

Advantages over Unguided Matching

Let w be image width in pixels, h be image height in pixels and $E\,d$ be feature density per pixel. Consider number of descriptor comparisons for brute force matching in unguided matching. This quantity amounts to $(E(d)wh)^2$. Now, consider a division of the image planes into regulary spaced grid, where each cell is of dimensions $\sqrt{w}\,\mathrm{x}\,\sqrt{h}$. Further, for each feature in the source image we know that it is located in one of the k of these cells in the target image. This means that under assumption that the image features are uniformly distributed across the destination image, the number of reqired comparisons ammounts to $E(dwh(kd\sqrt{w}\sqrt{h})) = dwh(kE(d)\sqrt{w}\sqrt{h})) < (E(d)wh)^2$ for small k.

1.3.5 On Assumtion of at Most One Image for each Landmark

1.4 Feature-Landmark Association

The topic of this section is design of Landmark Manager module. Purpose of this module threefold. First to maintain *feature-landmark association*. This task consists determining which features are observations of which landmark, given feature-feature association for edges in the pose-graph. Landmark Manager is invoked by Edge Builder (sec. XXX) after feature-feature association is done for

the edge that is beeing build. Second determining, if landmark can be triangulated precisely enough for the purposes of graph optimalization. Third triangulation of a landmark.

For given feature (f, v), consider a set of features L, such that $(f^{'}, v^{'}) \in L$, if and only iff there exist features $(f_1, v_1) = (f, v), (f_2, v_2) \cdots, (f_n, v_n) = (f^{'}, v^{'})$ with constraint such that (f_i, v_i) and (f_{i+1}, v_{i+1}) are feature-feature associated for $i = 0 \ldots n-1$. Such set L is called *feature track* and in a simplistic implementation of Landmark Manager, the set L could considered a set of feature observations of a landmark (i.e. be feature-landmark associated).

Feature-feature matches only satisfy epipolar constraint given by their respective nodes from which features were observed and the to features. This means that from a geometric standpoint, the features could be observations of diferent landmarks whre the two landmarks have to both lie in the epipolar plane given by the epipolar constraint. This can be further improved upon with the following idea. Triangulate the landmark from feature-feature pair and checking if those triangulations are consistent for a set of feature-feature pairs in a feature track. Such improvement is motivated by the fact that the optimalization method that we use (see sec XXX) utilizes feature-landmark associations and it is very sensistive to outliers.

In the discussion of the design of Landmark Manager is divided into three parts. First, the problem of determining the consistency of feature-feature association given a 3D estimate of landmark that is tested to be consistent with those two features in the association. Second, using the above building block, the method for determining landmark-feature association is presented. Third, the issue of deleting nodes in the pose-graph and the consequences of this for Landmark Manager are discussed.

1.4.1 Consistency Test for Feature-Feature Association

1.4.2 Feature-Landmark Association

1.4.3 Pose-Graph Node Deletion

1.5 Keyframe Management

1.6 Sliding Window Bundle Adjustment

We employ sliding window bundle adjustment to reduce drift as sugested in [FS12]. Bundle adjustment can be formulated as a non-linear lest squares problem on manifolds.

Introduction XXX.

1.6.1 non-liniear least square optimalization on manifolds

In the following, we define non-liniear least square optimalization on manifolds [Her08]. We have parameters $X = (x_1, x_2, ..., x_n) \in M$ that we wish to optimize. We are given set observations $Z = z_1, z_2, ..., z_m, z_i \in \mathcal{R}^{k_i}$, where the i^{th} observation is assumed to be determined by function $f_i \colon X \to \mathcal{R}^{k_i}$ up to a gausian noise.

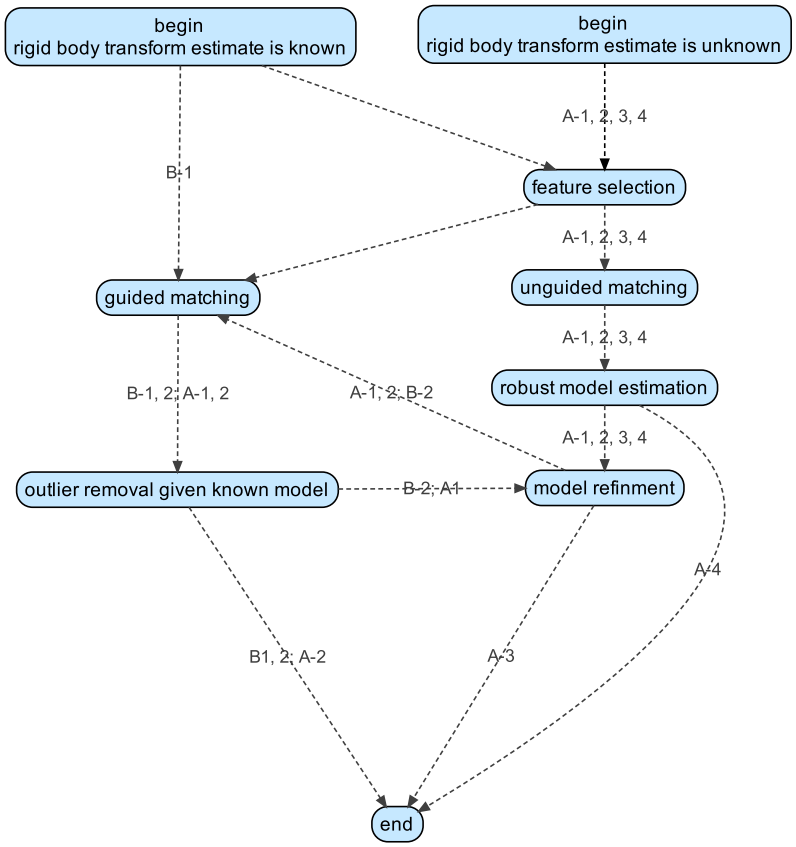


Figure 1.7: Progression of landmark states.

That is the measurement i is drawn from normal distribution with mean $f_i(X)$ and covariance matrix Σ (which is given) and the measurements are indepent given $f_i(X)$. This can be expressed by probability density $p_i(Z_i = z_i|X = x)$ which is called observation model in context of SLAM.

The goal is to determine maximum liklehood estimate

$$x^* = argmax_x(\prod_i p_i(Z_i, X = x) = argmin_x(\sum_i -log(p_i(Z_i|X = x))$$

Common algorithms that find solutions for the general form of the problem are Gauss-Newton, Levenberg-Marquardt and Conjugate Gradient methods. They proceed by starting with given inital estimate x_0 and in each iteration k deriving estimate x_{k+1} from estimate x_k of the previous iteration that has lower value of objective function. This means that on convergence x_k is only local optimum. Thus in order to find global optimum, the initial guess x_0, which is provided as an input should be close enough to the optimum.

Manifolds XXX and its increment operator XXX.

Euclidian spaces and minimal parametrizations XXX.

Error function XXX.

For introduction on how algorithms for non-liniear least square optimalization on manifolds (NLSOM) operate we refer the reader to [GKSB10] and [Her08].

Regarding our choice of optimalization framework, we have chosen *g2o* [KGS+11], because it is well documented [GKSK12], easy to use, well coded, compares itself favorably with its peers (XXXcitepees) in terms of efficiency and compared to

others, it is more general in terms of its interface and subclasses of nonlinear least squares optimalization problems that it is able to solve efficiently.

1.6.2 Optimalization Criteria for Our Problem

In this section we describe optimalization criteria used in context of *g2o*. In *g2o* NLMSOM is represented as a hypergraph, where nodes norespond to parameters x_i and hyperedges represent observation models in form of probability density $p_i(Z_i = z_i|X = x)$, where the hyperedge is connected to the nodes that p_i is dependent on. The probability density is specified in terms of function f_i and an information matrix $\Omega_i = \Sigma_i^{-1}$.

Node Parametrization We have two kinds of nodes – landmark nodes and camera pose nodes. Landmark nodes are represented as points in 3D space and parametrized as euclidian coordinates in $\mathcal{R}^n$ relative to world coordinate frame. Landmark increments have the same parametrization.

Camera poses are represented as rigid body transforms from world frame to coordinate frame of the camera. They are parametrized in the domain of $SE(3)$ group. The increments are parametrizied in domain of $se(3)$ group and the increment operator is realized by mapping increment around identity to its corresponding $SE(3)$ element using exponential map [MSKS10]. The resulting $SE3$ element is then multiplied by the computed $SE(3)$ element (rigid body transform concatenation). This is minimal parametrization (with respect to degrees of freedom) in eucleidian space. There are two common minimal parametrizations in euclidian spaces for increments – the $se3$ group and (tq) where t is translation vector and q is unit quaternion. We opted for the former because the two were experimentally evaluated in context of bundle adjustment in [KGS+11] and the $se3$ came out slightly better in terms of speed of convergence.

Observations Only pure observation in our task are observations of landmarks that are manifested by detected features in each camera pose (additionally estimated pose-to-pose transforms could be considered). Feature measurements are represented in image coordinates[3]. Problem of data association was already trated in section XXX. At this point in computation, the data associations are available. The covariance of information matrix is set to identity matrices. The gausian distribution of measurment is common assumtion in the literature in our setting as is independence of observation given model parameters.

Observation Model and Error Function As an observation model is obviously given by the projection of landmark corresponding to given observation (see section XXX for discussion of camera model). The error function adjusts for singularities at image borders[4]. This error function is commonly refered to as reprojection error.

Inicialization of Parameters

[3] should i switch to rays in 3d space???

[4] not yet

Choice of optimalization algorithm

1.6.3 Sliding Window BA

Bundle adjustment is what was discribed XXX. Explain windowed bundle adjustment XXX. Window size XXX. Which camera poses and landmarks are added into the optimalization and how their are computed XXX. Explain that optimalization of landmark poses is usful in the scale estimation XXX.

2. Experimental Results

2.1 Modes of Failure

2.2 Computational Requirement of the System and Its Components

Conclusion

Bibliography

[FS12] F. Fraundorfer and D. Scaramuzza. Visual odometry : Part ii: Matching, robustness, optimization, and applications. *Robotics Automation Magazine, IEEE*, 19(2):78–90, june 2012.

[GKSB10] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.

[GKSK12] G. Grisetti, R. Kummerle, H. Strasdat, and K. Konolige. g2o: A general framework for (hyper) graph optimization. jan. 2012. Last checked 2012.07.14.

[Her08] Christoph Hertzberg. A framework for sparse, non-linear least squares problems on manifolds, 2008. Supervisor: Udo Frese.

[KGS+11] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[KHK10] Jun-Sik Kim, Myung Hwangbo, and Takeo Kanade. Spherical approximation for multiple cameras in motion estimation: Its applicability and advantages. *Computer Vision and Image Understanding*, 114(10):1068 – 1083, 2010.

[LH06] Hongdong Li and R. Hartley. Five-point motion estimation made easy. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 630–633, 2006.

[MSKS10] Y. Ma, S. Soatto, J. Kosecká, and S.S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*, volume 26 of *Interdisciplinary Applied Mathematics Series*. Springer, New York, 2010.

[NNB06] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[SF11] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *Robotics Automation Magazine, IEEE*, 18(4):80–92, dec. 2011.

[SMD] H. Strasdat, JMM Montiel, and A.J. Davison. Scale drift-aware large scale monocular slam. bad bibtech file XXX.

[SRT+11] P. Sturm, S. Ramalingam, J.P. Tardif, S. Gasparini, and J. Barreto. *Camera Models and Fundamental Concepts Used in Geometric Computer Vision*, volume 6 of *Foundations and trends in computer graphics and vision*. Now Publishers, 2011.

[TPD08] J.P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *Intelligent*

Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 2531–2538. IEEE, 2008.

List of Tables

List of Abbreviations

VO	visual odometry
SLAM	simultaneous localization and mapping
VSLAM	visual simultaneous localization and mapping
BA	bundle adjustment
NLMSM	non-linear least mean squares on manifolds
RanSaC	random saple consensus
SVD	singular vector decomposition
CCM	central camera model
NCM	non-central camera model
CM	camera model
FOV	field of view

Attachments

Documentation

XXX.

ROS Node Dependencies

Bag-File Preparation

Other

The rest of the thesis may contain blank papers so that the thesis can be bound. The blank pages are not part of electronic version.